

REMARKS

Applicant respectfully requests reconsideration of the rejected claims. Claims 1-44 remain in the application. No claims have been amended or added.

Claim Rejections under 35 U.S.C. § 102(b)

Claims 1, 4, 5, 7, 13, 17-19, 21, 23, 28, 30, 32, 35, 36, and 38 have been rejected under 35 U.S.C. § 102(b) as being anticipated by Schneier, et al. "NPL Fast Software Encryption: Designing Encryption Algorithms for Optimal Software Speed on the Intel Pentium Processor" ("Schneier").

Claim 1

Applicant respectfully disagrees with the rejection because Schneier does not describe each and every element of the invention as claimed in claim 1. Applicant requests reconsideration of the rejected claim.

Claim 1 requires a method including receiving a data cipher operation and processing the data cipher operation. The processing includes "generating a number of portions of ciphertext from plaintext." The processing also includes "a load operation associated with the generating of at least **one portion** of the ciphertext" that "executes **prior** to a store operation associated with the generating of a **prior portion** of the ciphertext."

Schneier is basically an example of the prior art as described in Figure 1 of Applicants' application and explained in paragraphs [0006]-[0024]. Schneier describes the inner loop of an RC4 stream cipher that XORs 8 bits of keystream

data. This prior art algorithm, as described in Schneier and disclosed as the prior art in Applicants' application, loads a value $S[i]$ from the S-box array and assigns it to variable $tmpI$. The variable j is assigned the value of $j+tmpI$. Then, the prior art algorithm loads another value $S[j]$ from the S-box array and stores this value into $tmpJ$. Next, the prior art algorithm stores the values of the variables $tmpI$ and $tmpJ$ into the S-box array in locations $S[j]$ and $S[i]$ respectively. Following this, the prior art algorithm adds the values of $tmpI$ and $tmpJ$ and assigns this value to a variable t . Next, the prior art algorithm employs the value t as an index to reference an element in the S-box array. The prior art algorithm uses this t element of S-box array to XOR the 8 bit keystream into ciphered data.

Hence, the inner loop of this prior art algorithm only loads the values from the S-box needed to calculate the value of the cipher for one 8 bit keystream. In other words, the stores associated with one portion of 8 bit keystream are completed before a load associated with another portion of 8 bit keystream.

Schneier further describes the technique of unrolling this prior art algorithm to prevent a pipeline stall because of the address set up time required for table indexing. An example of unrolling an algorithm of the prior art is illustrated in Applicant's Figure 2. Yet, as Schneier describes, "the general performance problem with RC4 as designed is that almost every statement depends immediately on the statement before it, including the table index computation and the associated table accesses, limiting the amount of parallelism achievable." (Schneier, pg. 248). Moreover, not all of these dependencies can be untied even with unrolling the prior art algorithm. (Schneier, pg. 248). Furthermore, in this prior art algorithm "the

overlapping of the generation of the two different portions of cipher data are non-speculative in nature”; therefore, “to avoid the generation of inaccurate data for the ciphertext, the write operation to the S-box for the generation of a first portion of ciphertext is complete prior to the load operation to the S-box for the generation of a second portion of ciphertext.” (Application, para. [0024]). Thus, the stores associated with an 8 bit portion of keystream are completed before a load associated with another 8 bit portion of keystream.

Therefore, Schneier fails to describe a data ciphering operation as in claim 1 “wherein the processing comprises generating a number of portions of ciphertext from plaintext, wherein a load operation associated with the generating of at least **one portion** of the ciphertext executes prior to a store operation associated with the generating of a **prior portion** of the ciphertext.” By way of example, and not limitation, compare Figure 2 (Prior Art) to Figure 7.

In addition to failing to describe each and every element of claim 1 as discussed above, Schneier fails to render claim 1 obvious. Specifically, Schneier fails to suggest a process that generates a number of portions of ciphertext from plaintext, “wherein a load operation associated with the generating of at least **one portion** of the ciphertext executes prior to a store operation associated with the generating of a **prior portion** of the ciphertext.”

This aspect of claim 1 increases the number of process cycles that the memory is accessed over the prior art algorithm as disclosed in Schneier. Applicant’s claim 1 more fully utilizes the memory and provides for faster execution of data ciphering over that disclosed in Schneier. This aspect can be appreciated by

comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Because Schneier does not describe or suggest a data ciphering operation as in claim 1 “wherein the processing comprises generating a number of portions of ciphertext from plaintext, wherein a load operation associated with the generating of at least **one portion** of the ciphertext executes prior to a store operation associated with the generating of a prior portion of the ciphertext.” Schneier fails to render claim 1 obvious.

Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claim 1 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claims 4, 5, and 7

Applicant respectfully submits that claims 4, 5, and 7 depend on independent claim 1 and include all the limitations of claim 1. As such, claims 4, 5 and 7 are not anticipated by Schneier for at least the same reasons as claim 1.

Claim 13

Applicant respectfully disagrees with the rejection because Schneier does not teach each and every element of claim 13. Applicant requests reconsideration of the rejected claim.

Claim 13 requires a processing unit coupled to memory. The processing unit

executes a data ciphering operation. The processing unit "prior to the completion of the swapping of the data stored in the data structure for data ciphering of the **first portion** of the plaintext, . . . is to access data stored in the data structure for data ciphering of a **second portion** of the plaintext"

Schneier describes the inner loop of an RC4 stream cipher where stores associated with an 8 bit portion of keystream are completed before a load associated with another 8 bit portion of keystream, as discussed above. Thus, Schneier fails to describe a processing unit that "prior to the completion of the swapping of the data stored in the data structure for data ciphering of the **first portion** of the plaintext, . . . is to access data stored in the data structure for data ciphering of a **second portion** of the plaintext."

In addition to failing to describe each and every element of claim 13 as discussed above, Schneier fails to render claim 13 obvious. Specifically, Schneier does not suggest a processing unit that "prior to the completion of the swapping of the data stored in the data structure for data ciphering of the **first portion** of the plaintext, . . . is to access data stored in the data structure for data ciphering of a **second portion** of the plaintext"

This feature of claim 13 increases the number of process cycles that the memory is accessed over the prior art algorithm as described in Schneier. Applicant's claim 13 more fully utilizes the memory and provides for faster execution of data ciphering over that disclosed in Schneier. This aspect can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Because Schneier does not describe or suggest a processing unit that “**prior** to the completion of the swapping of the data stored in the data structure for data ciphering of the **first portion** of the plaintext, . . . is to access data stored in the data structure for data ciphering of a **second portion** of the plaintext,” Schneier fails to render claim 13 obvious.

Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claim 13 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claims 17-19

Applicant respectfully submits that claims 17-19 depend on independent claim 13 and include all the limitations of claim 13. As such, claims 17-19 are not anticipated by Schneier for at least the same reasons as claim 13.

Claim 21

Applicant respectfully disagrees with the rejection because Schneier does not teach each and every element of claim 21. Applicant requests reconsideration of the rejected claim.

Claim 21 requires a co-processor coupled to a host processor and a host memory. The co-processor includes an interface unit, and an execution unit. The execution unit includes memory, a microcontroller, and an RC4 unit. The RC4 unit “is to swap data . . . for data ciphering of a **first portion** of plaintext and . . . read

data . . . for data ciphering of a **second portion** of the plaintext, **prior to the completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plaintext.**"

Schneier describes the inner loop of an RC4 stream cipher where stores associated with an 8 bit portion of keystream are completed before a load associated with another 8 bit portion of keystream, as discussed above. Schneier fails to describe a co-processor including an RC4 unit that "is to swap data . . . for data ciphering of a **first portion** of plaintext and . . . read data . . . for data ciphering of a **second portion** of the plaintext, **prior to the completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plaintext.**" (claim 21).

In addition to failing to describe each and every element of claim 21 as discussed above, Schneier fails to render 21 obvious. Specifically, Schneier does not suggest a co-processor including an RC4 unit that "is to swap data . . . for data ciphering of a **first portion** of plaintext and . . . read data . . . for data ciphering of a **second portion** of the plaintext, **prior to the completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plaintext.**" (claim 21).

This aspect of claim 21 increases the number of process cycles that the memory is accessed over the prior art algorithm as disclosed in Schneier. Applicant's claim 21 more fully utilizes the memory and provides for faster execution of data ciphering over that disclosed in Schneier. This aspect can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Because Schneier does not describe or suggest a co-processor including an

RC4 unit that “is to swap data . . . for data ciphering of a **first portion** of plaintext and . . . read data . . . for data ciphering of a **second portion** of the plaintext, prior to the completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plaintext,” Schneier fails to render claim 21 obvious.

Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claim 21 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claim 23

Applicant respectfully submits that claim 23 depends on independent claim 21 and includes all the limitations of claim 21. As such, claim 23 is not anticipated by Schneier for at least the same reasons as claim 21.

Claim 28

Applicant respectfully disagrees with the rejection because Schneier does not teach each and every element of claim 28. Applicant requests reconsideration of the rejected claim.

Claim 28 requires a system including a host processor, a host memory, and a co-processor. The co-processor includes an interface unit, and an execution unit. The execution unit includes a memory, a microcontroller, and an RC4 unit. The RC4 unit “is to swap data stored . . . for data ciphering of a **first portion** of the plaintext and . . . is to read data stored . . . for data ciphering of a **second portion** of the

plaintext, prior to completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plain text.”

Schneier describes the inner loop of an RC4 stream cipher algorithm where stores associated with an 8 bit portion of keystream are completed before a load associated with another 8 bit portion of keystream, as discussed above. Schneier fails to describe a co-processor including an RC4 unit that “is to swap data stored . . . for data ciphering of a **first portion** of the plaintext and . . . is to read data stored . . . for data ciphering of a **second portion** of the plaintext, prior to completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plain text.” (claim 28).

In addition to failing to describe each and every element of claim 28 as discussed above, Schneier fails to suggest a co-processor including an RC4 unit that “is to swap data stored . . . for data ciphering of a **first portion** of the plaintext and . . . is to read data stored . . . for data ciphering of a **second portion** of the plaintext, prior to completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plain text.” (claim 28).

This aspect of claim 28 increases the number of process cycles that the memory is accessed over the prior art algorithm as described in Schneier. Applicant’s claim 28 more fully utilizes the memory and provides for faster execution of data ciphering over that described in Schneier. This aspect can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Because Schneier does not describe a co-processor including an RC4 unit

that “is to swap data stored . . . for data ciphering of a **first portion** of the plaintext and . . . is to read data stored . . . for data ciphering of a **second portion** of the plaintext, prior to completion of the swapping of data stored . . . for data ciphering of the **first portion** of the plain text,” Schneier fails to render claim 28 obvious.

Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claim 28 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claim 30

Applicant respectfully submits that claim 30 depends on independent claim 28 and includes all the limitations of claim 28. As such, claim 30 is not anticipated by Schneier for at least the same reasons as claim 28.

Claim 32

Applicant respectfully disagrees with the rejection because Schneier does not teach each and every element of claim 32. As such, Applicant requests reconsideration of the rejected claim.

Claim 32 requires a machine-readable medium that provides instructions including receiving a data cipher operation and processing the data cipher operation. The processing includes “generating a number of portions of ciphertext from plaintext” such that “a load operation associated with the generating of at least **one portion** of the ciphertext executes prior to a store operation associated with the

generating of a **prior portion** of the ciphertext.”

As discussed above, Schneier describes an inner loop of an RC4 stream cipher where stores associated with an 8 bit portion of keystream are completed before a load associated with another 8 bit portion of keystream. Schneier fails to describes processing that includes “generating a number of portions of ciphertext from plaintext” such that “a load operation associated with the generating of at least **one portion** of the ciphertext executes **prior** to a store operation associated with the generating of a **prior portion** of the ciphertext.” (claim 32).

In addition to failing to describe each and every element of claim 32 as discussed above, Schneier fails to render claim 32 obvious. Specifically, Schneier fails to suggest processing that includes “generating a number of portions of ciphertext from plaintext” such that “a load operation associated with the generating of at least **one portion** of the ciphertext executes **prior** to a store operation associated with the generating of a **prior portion** of the ciphertext.” (claim 32).

This aspect of claim 32 increases the number of process cycles that the memory is accessed over the prior art algorithm as disclosed in Schneier. Applicant’s claim 32 more fully utilizes the memory and provides for faster execution of data ciphering over that disclosed in Schneier. This aspect can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Because Schneier does not disclose or suggest processing that includes “generating a number of portions of ciphertext from plaintext” such that “a load operation associated with the generating of at least **one portion** of the ciphertext

executes **prior** to a store operation associated with the generating of a **prior portion**
of the ciphertext,” Schneier fails to render claim 32 obvious.

Applicant is unaware of any knowledge in the art of speculative execution for data ciphering operations that in view of Schneier would render claim 1 obvious. Applicant respectfully requests a reference be identified so Applicant may directly address the combination.

Claims 35, 36, and 38

Applicant respectfully submits that claims 35, 36, and 38 depend on independent claim 32 and include all the limitations of claim 32. As such, claims 35, 36, and 38 are not anticipated by Schneier for at least the same reasons as claim 32.

Claim Rejections under 35 U.S.C. §103(a)

Claims 2, 3, 15, 16, 20, 33, and 34

Claims 2, 3, 15, 16, 20, 33, and 34 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Schneier.

Claims 2 and 3

Applicant respectfully submits that claims 2 and 3 depend on independent claim 1 and include all the limitations of claim 1. As such, claims 2 and 3 are patentable for at least the same reasons as claim 1.

Claims 15, 16 and 20

Applicant respectfully submits that claims 15, 16, and 20 depend on independent claim 13 and include all the limitations of claim 13. As such, claim 15, 16 and 20 are patentable for at least the same reasons as claim 13.

Claims 33 and 34

Applicant respectfully submits that claims 33 and 34 depend on independent claim 32 and include all the limitations of claim 32. As such, claims 33 and 34 are patentable for at least the same reasons as claim 32.

Claims 6, 8-12, 14, 22, 24, 29, 31, 37, and 39-44

Claims 6, 8-12, 14, 22, 24, 29, 31, 37, and 39-44 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Schneier and further in view of Puziol et al., U.S. Patent No. 5,454,117 ("Puziol").

Claim 6

Applicant respectfully submits that claim 6 depends on independent claim 1 and includes all the limitations of claim 1. As such, claim 6 is patentable for at least the same reasons as claim 1.

In addition, Puziol describes configurable branch prediction hardware for a processor, comprising logic and interconnect, which is configurable via a control line. The branch prediction logic is made up of branch prediction cache, next state logic, predicted direction logic, write address generator, history ram, and read address

generator. This hardware is used to predict the target address of the next set of target instructions for the processor to prevent pipeline bubbles in a processor.

The hardware makes a static and dynamic prediction for each branch. The static prediction is based on the branch's opcode. Unconditional branches have a static prediction of taken. Conditional branches have static prediction of not-taken. Prediction direction logic is used to resolve if a conditional loop instruction is given a prediction of taken or not-taken. The dynamic predictions are made at run time. In the case the branch prediction logic mispredicts the target address, the "correct" instruction bytes are read from the branch prediction cache and the mispredicted address is aborted.

Furthermore, branch predictors, such as Puziol, determine the proper address for the next set of instructions prior to the processor making this request to optimize the efficiency of a processor. Branch predictors do not move loads in front of stores to optimize RC4 algorithms.

Because Schneier and Puziol do not describe a data ciphering operation "wherein the processing comprises generating a number of portions of ciphertext from plaintext, wherein a load operation associated with the generating of at least one portion of the ciphertext executes prior to a store operation associated with the generating of a prior portion of the ciphertext" including aborting a generated portion of ciphertext "upon determining that the data being swapped equals the data being accessed in the data structure," the combination fails to render claim 6 obvious.

Claim 8

Applicant respectfully disagrees with the rejection because the combination does not teach each and every element of the invention as claimed in claim 8.

Applicants request reconsideration of the rejected claim.

Claim 8 requires a computer-implemented method that includes receiving a request to perform data ciphering of plaintext and then processing the request. The processing of the data ciphering of plaintext includes "data ciphering a **first portion** of the plaintext based on" swapped data from a first access of data. The processing further includes "performing a **second access of data** from the data structure **prior to the swapping of the data from the **first access****." "Upon determining that the data from the first access does not equal the data from the second access," the processing includes "data ciphering a **second portion** of the plaintext based on swapped data from the second access."

As discussed above, Schneier describes an inner loop of an RC4 stream cipher where stores associated with an 8 bit portion of keystream are completed before a load associated with another 8 bit portion of keystream. Schneier fails to describe processing that ciphers a first portion of plaintext that includes "performing a **second access of data** from the data structure **prior to the swapping of the data from the **first access****" and that "upon determining that the data from the first access does not equal the data from the second access, . . . data ciphering a second portion of the plaintext based on swapped data from the second access." (claim 8).

This aspect of claim 8 increases the number of process cycles that the memory is accessed over the prior art algorithm as described in Schneier.

Applicant's claim 8 more fully utilizes the memory and provides for faster execution of data ciphering over that described in Schneier. This aspect can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

As discussed above, Puziol describes configurable branch prediction hardware for a processor used to predict the target address of the next set of target instructions for the processor to prevent pipeline bubbles in a processor. Puziol fails to describe processing that ciphers a first portion of plaintext that includes "performing a **second access** of data from the data structure **prior to the swapping of the data from the **first access**"** and that "upon determining that the data from the first access does not equal the data from the second access, . . . data ciphering a second portion of the plaintext based on swapped data from the second access." (claim 8).

This aspect of claim 8 increases the number of process cycles that the memory is accessed over the prior art algorithm as described in Schneier. Applicant's claim 8 more fully utilizes the memory and provides for faster execution of data ciphering over that described in Schneier. This aspect can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Furthermore, branch predictors, such as Puziol, determine the proper address for the next set of instructions prior to the processor making this request to optimize the efficiency of a processor. Branch predictors do not move loads in front of stores to optimize RC4 algorithms.

Because Schneier and Puziol do not describe or suggest processing that ciphers a first portion of plaintext that includes “performing a **second access** of data from the data structure **prior to the swapping of the data from the first access**” and that “upon determining that the data from the first access does not equal the data from the second access, . . . data ciphering a second portion of the plaintext based on swapped data from the second access,” the combination of Schneier and Puziol fails to render claim 8 obvious.

Claims 9-12

Applicant respectfully submits that claims 9-12 depend on independent claim 8 and include all the limitations of claim 8. As such, the combination of Schneier and Puziol fails to render claims 9-12 obvious for at least the same reasons as claim 8.

Claim 14

Applicant respectfully submits that claim 14 depends on independent claim 13 and includes all the limitations of claim 13. As such, claim 14 is patentable for at least the same reasons as claim 13.

Claims 22 and 24

Applicant respectfully submits that claims 22 and 24 depend on independent claim 21 and include all the limitations of claim 21. As such, the combination of Schneier and Puziol fails to render claims 22 and 24 obvious for at least the same reasons as claim 21.

Claims 29 and 31

Applicant respectfully submits that claims 29 and 31 depend on independent claim 28 and include all the limitations of claim 28. As such, the combination of Schneier and Puziol fails to render claims 29 and 31 obvious for at least the same reasons as claim 28.

Claim 37

Applicant respectfully submits that claim 37 depends on independent claim 32 and includes all the limitations of claim 32. As such, the combination of Schneier and Puziol fails to render claim 37 obvious for at least the same reasons as claim 32.

Claims 39

Applicant respectfully disagrees with the rejection because the combination does not teach each and every element of the invention as claimed in claim 39. Applicant requests reconsideration of the rejected claim.

Claim 39 requires a machine-readable medium that provides instructions, which when executed by a machine, cause operations including receiving a request to perform data ciphering of plaintext and then processing the request. The processing of the data ciphering of plaintext includes "data ciphering a **first portion** of the plaintext based on" swapped data from a first access of data. The processing further includes "performing a **second access** of data from the data structure prior

to the swapping of the data from the **first access**.” “Upon determining that the data from the first access does not equal the data from the second access,” the processing further includes “data ciphering a **second portion** of the plaintext based on swapped data from the second access.”

As discussed above, Schneier describes an inner loop of an RC4 stream cipher where stores associated with an 8 bit portion of keystream are completed before a load associated with another 8 bit portion of keystream. Schneier fails to describe processing that ciphers a first portion of plaintext that includes “performing a **second access** of data from the data structure prior to the swapping of the data from the **first access**” and “upon determining that the data from the first access does not equal the data from the second access, . . . data ciphering a **second portion** of the plaintext based on swapped data from the second access.” (claim 39).

This aspect of claim 39 increases the number of process cycles that the memory is accessed over the prior art algorithm as described in Schneier. Applicant's claim 39 more fully utilizes the memory and provides for faster execution of data ciphering over that described in Schneier. This aspect can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

As discussed above, Puziol describes configurable branch prediction hardware for a processor used to predict the target address of the next set of target instructions for the processor to prevent pipeline bubbles in a processor. Puziol fails to describe processing that ciphers a first portion of plaintext that includes

“performing a **second access** of data from the data structure **prior to the swapping of the data from the first access**” and “upon determining that the data from the first access does not equal the data from the second access, . . . **data ciphering a second portion** of the plaintext based on swapped data from the second access.” (claim 39).

This aspect of claim 39 increases the number of process cycles that the memory is accessed over the prior art algorithm as disclosed in Puziol. Applicant’s claim 39 more fully utilizes the memory and provides for faster execution of data ciphering over that disclosed in Puziol. This aspect can be appreciated by comparing Figure 2, the prior art algorithm, with Figure 7, an example of an embodiment and not provided as a limitation thereof.

Furthermore, branch predictors, such as Puziol, determine the proper address for the next set of instructions prior to the processor making this request to optimize the efficiency of a processor. Branch predictors do not move loads in front of stores to optimize RC4 algorithms.

Because Schneier and Puziol do not describe processing that ciphers a first portion of plaintext that includes “performing a **second access** of data from the data structure **prior to the swapping of the data from the first access**” and “upon determining that the data from the first access does not equal the data from the second access, . . . **data ciphering a second portion** of the plaintext based on swapped data from the second access,” the combination of Schneier and Puziol fails to render obvious claim 39.

Claim 40-44

Applicants respectfully submit that claims 40-44 depend on independent claim 39 and include all the limitations of claim 39. As such, the combination of Schneier and Puziol fails to render claims 40-44 obvious for at least the same reasons as claim 39.

Claims 25-27

Claims 25-27 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Batcher, U.S. Patent No. 6,873,707 ("Batcher") and further in view of Puziol.

Claim 25

Applicant respectfully disagrees with the rejection because the combination does not teach each and every element of the invention as claimed in claim 25, Applicant requests reconsideration of the rejected claim.

Claim 25 requires an apparatus including a memory and an RC4 hardware state machine. The two are coupled together "to generate a plurality of output text blocks from a plurality of input text blocks." The RC4 state machine generates "a subset of said plurality of output text blocks . . . as a result of repeating the same sequence of states." During these "sequence of states data is **speculatively read** . . . as part of the generation of a **next one** of said plurality of output text blocks **prior to a write . . . completing as part of generation of a **current one** of said plurality of output text blocks.**"

Batcher describes a cycle stealing hardware configuration that contributes to faster processing of the S-box table initialization phase and the decrypt/encrypt phase of an RC4 algorithm. Batcher accelerates operations of a microcode controller by using a level sensitive address latch and a level sensitive instruction code word latch to create the effect of a dual ported microcode storage, which allows the fetch of the next microcode word and current microcode execute operation to proceed on the same clock cycle. This configuration allows the clock frequency of the system to be increased by twice the frequency as the prior art because of the setup/hold relationship to the latches are improved. (Batcher, col. 10, ll. 12 – 15).

That is this cycle stealing hardware configuration uses the leading and falling edge of a clock signal to speed up the data accessing. However, the hardware configuration does not “speculatively read . . . as part of the generation of a next one of said plurality of output text blocks prior to a write . . . completing as part of generation of a current one of said plurality of output text blocks.” (claim 25). Furthermore, cycle stealing hardware configurations, such as Batcher, do not move loads in front of stores to optimize RC4 algorithms.

As discussed above, Puziol describes configurable branch prediction hardware for a processor used to predict the target address of the next set of target instructions for the processor to prevent pipeline bubbles in a processor. Puziol fails to describe an RC4 state machine that generates “a subset of said plurality of output text blocks . . . as a result of repeating the same sequence of states” in which “data is speculatively read . . . as part of the generation of a next one of said plurality of

output text blocks prior to a write . . . completing as part of generation of a **current one** of said plurality of output text blocks.” (claim 25).

Furthermore, branch predictors, such as Puziol, determine the proper address for the next set of instructions prior to the processor making this request to optimize the efficiency of a processor. Branch predictors do not move loads in front of stores to optimize RC4 algorithms.

Because Batchner and Puziol do not describe an RC4 state machine that generates “a subset of said plurality of output text blocks . . . as a result of repeating the same sequence of states” in which “data is speculatively read . . . as part of the generation of a next one of said plurality of output text blocks prior to a write . . . completing as part of generation of a current one of said plurality of output text blocks,” the combination of Batchner and Puziol fails to render claim 25 obvious.

Claim 26 and 27

Applicant respectfully submits that claims 26 and 27 depend on independent claim 25 and include all the limitations of claim 25. As such, the combination of Batchner and Puziol fails to render claims 26 and 27 obvious for at least the same reasons as claim 25.

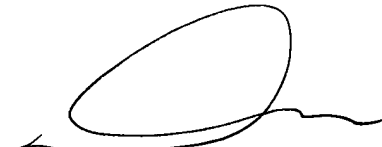
Conclusion

Applicant respectfully submits that the rejections have been overcome by the remarks. Accordingly, Applicant respectfully requests the rejections be withdrawn and the claims allowed. If the allowance of these claims could be facilitated by a telephone conference, the Examiner is invited to contact the undersigned at (408) 720-8300. If there are any additional charges, please charge our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: 11/2, 2005



Daniel M. De Vos
Registration No. 37,813

Customer No. 08791
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1030
(408) 720-8300